Project 9: Java Graphics

Project Objectives

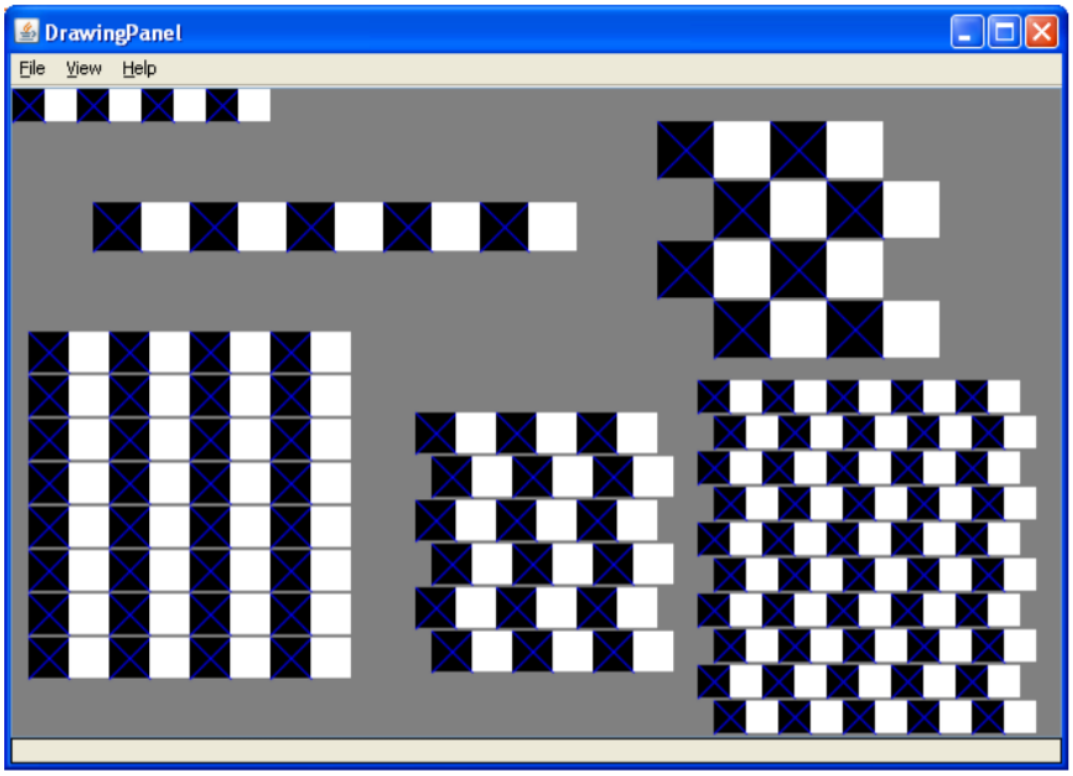
In this project, you will

- Practice Java graphics call
- Use methods provided by other Java classes/libraries
- Practice loops, value parameters, etc.

Exercise 1: CafeWall.java

Problem Statement:

In this exercise, we will be exploring something that is known as Café Wall illusion (as described in https://en.wikipedia.org/wiki/Caf%C3%A9_wall_illusion). You will turn in a file "CafeWall.java", which produce the image below. The DrawingPanel has a size of 650 pixels by 400 pixels and a gray background.



This image has several levels of structure. Black and white squares are used to form rows, and rows are combined to form grids. The output has two free-standing rows and four grids. You should first write a method to produce a single row. Each row is composed of a certain number of black/white pairs of boxes where each black box has a blue X in it. You should then write another method that uses your row method to produce a single grid.

Suggestions on Development Strategy (How to get started):

1. Draw a row

   First, it would be better to start with drawing a row. A row should have the following properties: start position (x, y), number of box pairs, size of each box. The start position is the coordinates of the top-left corner of the row. (You can also add some additional fields so that user can change the start position to middle-left, bottom-left, etc.) The boxes are specified using a single parameter because each should be a square. You can use one or more for loops to write this code so that it can provide any of the various rows.

2. Draw a grid

   Once you have completed the "draw a row" functionality, write a method that produces a grid of these rows by calling your row method appropriately (you will again use one or more for loops to solve this task). Grids are composed of a series of pairs of rows where the second row is offset a certain distance in the x direction relative to the first. The output has four grids with the following properties: start position (x, y), number of row pairs, size of each box, even-row offset.

   Each grid is a square, which is why a single value (number of pairs) indicates the number of rows and columns. For example, as the table above indicates, the lower-left grid is composed of 4 pairs. That means each row has four pairs of black/white boxes (8 boxes in all) and the grid itself is composed of 4 pairs of rows (8 rows total). A single box size is again used because each box should be a square. The offset indicates how far the second row should be shifted to the right in each pair. The figure in the lower left has no offset at all. The grid in the upper-right is offset by the size of one of the boxes, which is why it has a checkerboard appearance. The other two grids have an offset that is in between, which is why they produce the optical illusion (the rows appear not to be straight even though they are).

   Each pair of lines in the grid should be separated by a certain distance, revealing the gray background underneath. This is referred to as the "mortar" that would appear between layers of brick if you were to build a wall with this pattern. The mortar is essential to the illusion. Your program should use 2 pixels of separation for the mortar, but you must introduce a class constant that would make it easy to change this value to something else.

Style Guidline

You are required to have the two methods described above (one for a single row, one for a grid). You may use additional methods if you like. You are expected to use parameters effectively, and none of your methods should take unnecessary parameters. For this assignment, a parameter is considered unnecessary if its value is redundant with another parameter, if its value could be computed using other parameters' values, or if it is never used in the method. We expect you to use good style throughout your code. Give meaningful names to your methods, variables, and parameters. Properly indent your code. Follow Java's naming conventions as specified in Chapter 1. Limit the lengths of your lines to fewer than 100 characters. Include meaningful header comments at the top of your program and at the start of each method.